Składowane programy PL/SQL

- Przechowywane trwale w bazie danych w postaci skompilowanej
- Wykonywane na żądanie użytkownika lub zajścia określonych zdarzeń
- Mogą być współdzielone przez wielu użytkowników

  – procedury – wykonują określone akcje,
  – funkcje – wykonują obliczenia i zwracają wartości,
  – pakiety – biblioteki procedur i funkcji,
  – wyzwalacze – procedury wywoływane automatycznie przez zdarzenia

Parametry programów

- Umożliwiają przekazanie wartości do wnętrza programu, a także wartości z wnętrza programu do środowiska wołającego
- Parametr formalny – używany w deklaracji programu
- Parametr aktualny – podawany przy wywołaniu programu

Deklaracja parametru formalnego:

nazwa [tryb_przekazania] typ [**DEFAULT** wartość_domyślna]

Nie podaje się długości typu dla parametru formalnego

Parametry programów

Tryby przekazania

IN	OUT	IN OUT
Tryb domyślny	**Musi być określony**	**Musi być określony**
Przekazuje wartość do programu ze środowiska wołającego	Przekazuje wartość z programu do środowiska wołającego	Przekazuje wartość ze środowiska wołającego do programu i z programu do środowiska wołającego
Parametr formalny w programie zachowuje się jak stała, nie można przypisywać mu wartości	Parametr formalny w programie zachowuje się jak nie zainicjalizowana zmienna	Parametr formalny w programie zachowuje się jak zainicjalizowana zmienna
Parametr aktualny może być literałem, wyrażeniem, stałą lub zmienną	Parametr aktualny musi być zmienną	Parametr aktualny musi być zmienną

PL/SQL - procedury

- Wykonuje określone akcje

Tworzenie procedury:

```
CREATE [OR REPLACE] PROCEDURE nazwa_procedury
[(lista parametrów)] IS
<sekcja deklaracji stałych, zmiennych, wyjątków i kursorów>
BEGIN
<ciało procedury>
END [nazwa_procedury];
```

- Sekcja deklaracji – między IS a BEGIN (bez DECLARE)
- Opcjonalna sekcja obsługi wyjątków – jako ostatnia sekcja przed końcem procedury

PL/SQL - procedury

```
CREATE PROCEDURE SprawdzSamochod
   (p_nr_rej VARCHAR2 DEFAULT null) IS
   v_marka samochody.marka%TYPE;
   v_model samochody.model%TYPE;

BEGIN
   IF p_nr_rej is not null THEN
       SELECT marka, model INTO v_marka, v_model FROM samochody
        WHERE nr_rej = p_nr_rej;
   END IF;
 dbms_output.put_line(' Dane samochodu o numerze rejestracyjnym
'||p_nr_rej||' marka '||v_marka||' model '||v_model);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
     raise_application_error (-20001, 'Niepoprawny numer
     rejestracyjny samochodu !!!');
END SprawdzSamochod;
```

PL/SQL - procedury

- Wywołanie procedury:
  – z innego programu PL/SQL lub anonimowego bloku PL/SQL

```
BEGIN
SprawdzSamochod('KR12345R');
SprawdzSamochod('WA12345L');
END;
```

  – z narzędzia SQL*Plus

```
SQL> execute SprawdzSamochod('KR12345R');
```

PL/SQL - przykład

Napisz procedurę Podwyzka, która wszystkim samochodom danej marki (parametr) podniesie koszt dzienny wypożyczenia o podany procent (parametr) - domyślnie podwyżka powinna wynosić 10%

```
CREATE OR REPLACE PROCEDURE Podwyzka
   (p_marka IN VARCHAR2, p_procent IN NUMBER DEFAULT 10) IS
BEGIN
UPDATE samochody SET koszt_dnia = koszt_dnia * (1 + p_procent/100)
WHERE marka = p_marka;
END Podwyzka;
```

PL/SQL - funkcje

- Wykonuje obliczenia i zwraca wartość do środowiska wołającego.

Tworzenie funkcji:

CREATE [OR REPLACE] FUNCTION nazwa_funkcji
[(lista parametrów)] **RETURN** typ_zwracanej_wartości **IS**
<sekcja deklaracji stałych, zmiennych, wyjątków i kursorów>
BEGIN
<ciało funkcji>
END [nazwa_funkcji];

- W ciele funkcji musi wystąpić polecenie RETURN <wyrażenie>, kończące działanie funkcji i zwracające wartość wyrażenia do środowiska wołającego

PL/SQL - funkcje

```
CREATE FUNCTION LiczbaSamochodow
(p_marka IN VARCHAR2 DEFAULT null) RETURN NUMBER  IS
v_liczba_samochodow NUMBER(5);
BEGIN
IF p_marka is null THEN
SELECT count(*) INTO v_liczba_samochodow FROM samochody;
ELSE
SELECT count(*) INTO v_liczba_samochodow FROM samochody
WHERE marka = p_marka;
END IF;
RETURN v_liczba_samochodow;
END LiczbaSamochodów;
```

PL/SQL - funkcje

- Wywołanie funkcji:
  - z innego programu PL/SQL lub anonimowego bloku:

```
DECLARE
v_sam_marki NUMBER(5);
v_marka samochody.marka%TYPE := 'FORD';
BEGIN
v_sam_marki := LiczbaSamochodow(v_marka);
dbms_output.put_line('Mamy samochodow marki FORD: ' ||
to_char(v_sam_marki));
END;
```

  - z polecenia SQL

```
SELECT LiczbaSamochodow(marka)
FROM Samochody;
```

PL/SQL - przykład

Napisz funkcję KosztNetto, która dla podanej kwoty brutto (parametr) i podanej stawki podatku (parametr o wartości domyślnej 10%) wyliczy koszt netto

```
CREATE OR REPLACE FUNCTION KosztNetto
(p_koszt_brutto IN NUMBER, p_stawka IN NUMBER DEFAULT 10)
RETURN NUMBER IS
v_koszt_netto NUMBER(10,2);
BEGIN
v_koszt_netto := p_koszt_brutto * (1 - p_stawka/100);
RETURN v_koszt_netto;
END KosztNetto;
```

PL/SQL - pakiety

- Biblioteka procedur i funkcji.
- Składa się z dwóch części:
  - specyfikacji (interfejs)
  - ciała (implementacja)
- Użytkownik ma możliwość wywołania tylko tych procedur i funkcji, które są zadeklarowane w specyfikacji
- Implementacja może zostać ukryta przed użytkownikiem

Kroki tworzenia pakietu:

1. utworzenie specyfikacji
2. utworzenie ciała (opcjonalne)

PL/SQL - pakiety

CREATE [OR REPLACE] PACKAGE nazwa_pakietu **IS**
<deklaracje stałych, zmiennych, kursorów i wyjątków dostępnych dla użytkowników pakietu>
<deklaracje procedur i funkcji, dostępnych dla użytkowników pakietu>
END [nazwa_pakietu];

specyfikacja

CREATE [OR REPLACE] PACKAGE BODY nazwa_pakietu **IS**
<deklaracje stałych, zmiennych, kursorów i wyjątków dostępnych tylko dla programów wewnątrz pakietu>
<definicje procedur i funkcji, zadeklarowanych w specyfikacji pakietu>
<definicje procedur i funkcji, dostępnych dla programów wewnątrz pakietu>
END [nazwa_pakietu];

ciało

PL/SQL - pakiety

```
CREATE PACKAGE Samochod IS
PROCEDURE SprawdzSamochod (p_marka IN varchar2);
FUNCTION LiczbaSamochodow(p_marka IN varchar2) RETURN
number;
END Samochod;
```

```
CREATE PACKAGE BODY Samochod IS
PROCEDURE SprawdzSamochod(p_marka IN varchar2) IS
...
END SprawdzSamochod;
FUNCTION LiczbaSamochodow(p_marka IN VARCHAR2 DEFAULT null)
RETURN number IS
...
END LiczbaSamochodow;
FUNCTION KosztNetto(p_koszt_brutto IN NUMBER, p_stawka IN
NUMBER DEFAULT 10) RETURN number  IS
...
END KosztNetto ;
END Samochod;
```

PL/SQL - pakiety

- Wywoływanie procedur i funkcji z pakietu – te same zasady co dla zwykłych procedur i funkcji, nazwa programu poprzedzona nazwą pakietu

```
DECLARE
v_liczba_samochodow NUMBER(5);
v_marka samochody.marka%TYPE := 'FORD';
BEGIN
Samochod.SprawdzSamochod('KR12345R');
v_liczba_samochodow := Samochod.LiczbaSamochodow (v_marka);
END;
```

```
SELECT Samochod.LiczbaSamochodow(v_marka)
FROM Samochody;
```

PL/SQL - pakiety

Napisz pakiet Konwersja, zawierający funkcje: konwertującą skalę Celsjusza na skalę Fahrenheita) oraz funkcję konwertującą skalę Fahrenheita na skalę Celsjusza

Wskazówka:

$$TC = 5/9 * (TF - 32)$$

$$TF = 9/5 * TC + 32$$

PL/SQL - pakiety

- Wywoływanie procedur i funkcji z pakietu – te same zasady co dla zwykłych procedur i funkcji, nazwa programu poprzedzona nazwą pakietu

```
CREATE OR REPLACE PACKAGE Konwersja IS
FUNCTION CnaF(p_celsjusz IN NUMBER) RETURN NUMBER;
FUNCTION FnaC(p_fahrenheit IN NUMBER) RETURN NUMBER;
END Konwersja;
```

PL/SQL - ciało

```
CREATE OR REPLACE PACKAGE BODY Konwersja IS
  FUNCTION CnaF(p_celsjusz IN NUMBER) RETURN   NUMBER IS
  BEGIN
     RETURN p_celsjusz * 9/5 + 32;
  END CnaF;
  FUNCTION FnaC(p_fahrenheit IN NUMBER) RETURN NUMBER IS
  BEGIN
    RETURN (p_fahrenheit - 32) * 5/9;
  END FnaC;
END Konwersja;
```

Sekwencje

- Sekwencje w bazie danych są obiektami upraszczającymi proces tworzenia jednoznacznych identyfikatorów rekordów w bazie
- Sekwencją nazywa sie automatyczny licznik generujący nową wartość liczbową przy każdym jej wywołaniu
- Stosowanie sekwencji pozwala na generowanie niepowtarzalnych numerów, co upraszcza tworzenie kluczy głównych o wartosciach unikatowych

CREATE SEQUENCE nazwa sekwencji

[**START WITH** liczba całkowita] – deklaruje wartość liczbową od której ma zaczynać sie numeracja

[**INCREMENT BY** liczba całkowita] – o ile zwiększona będzie wartość sekwencji przy każdym kolejnym jej wywołaniu

[**MINVALUE** liczba całkowita]

[**MAXVALUE** liczba całkowita]

Sekwencje

```
CREATE SEQUENCE NUMERACJA
START WITH 10
INCREMENT BY 5
MAXVALUE 100;
```

tworzy sekwencję rozpoczynającą się od 10, której każda kolejna wartość, większa jest od poprzedniej 0 5, max wartość sekwencji może wynosić 100

Z mechanizmem sekwencji związane są dwie pseudokolumny

- **CURRVAL** – zwraca aktualną wartość sekwencji
- **NEXTVAL** – tworzy kolejną wartość sekwencji i zwraca ją

Sekwencje

Sprawdzenie aktualnej wartości sekwencji:

```
SELECT NUMERACJA.CURRVAL
FROM DUAL
```

Generacja i zwrot kolejnej wartości sekwencji:

```
SELECT NUMERACJA.NEXTVAL
FROM DUAL
```

- Tabela DUAL jest tworzona automatycznie przez bazę danych w ramach słownika danych
- Znajduje się ona w schemacie użytkownika SYS, ale jest dostępna pod nazwą DUAL dla wszystkich pozostałych użytkowników

Sekwencje

- używając generatora sekwencji, dane do bazy wstawia się w następujący sposób:

INSERT INTO KLIENCI(ID_KLI.NEXTVALL, ...

do modyfikacji sekwencji służy instrukcja
ALTER SEQUENCE
pozwala ona na zmianę parametrów sekwencji oprócz wartości początkowej

Aby zmienić wartość początkową trzeba usunąć sekwencję
DROP SEQUENCE
i utworzyć ją od nowa

Sekwencje

- Informacje o utworzonych w schemacie użytkownika sekwencjach można uzyskać z perspektywy:

  USER_SEQUENCE słownika danych

```
SELECT SEQUENCE_name, MIN_VALUE,
MAX_VALUE,INCREMENT_BY  FROM USER_SEQUENCE
```